INFO-F101 : Programmation
Syllabus d'exercices

Réalisé par l'équipe d'assistants du cours INFO-F101

Département d'Informatique, Faculté des Sciences
Université libre de Bruxelles

Édition 2012–2013
« Crunchy Frog »

Remerciements et historique

Édition 2012–2013 (réorganisation des exercices, création d'exercices préparatoires, passage à Python 3). Liran Lerman, Markus Lindström (coordinateur du chantier), Luciano Porretta et Nikita Veshchikov. Merci également à Marie Boulvain (conseillère pédagogique du Centre de Didactique Supérieure de l'Académie universitaire Wallonie-Bruxelles) pour l'aide à la rédaction de la préface.

Édition 2011–2012 (création chapitre « Instructions de base » et compléments sur complexité et invariants). Stéphane Fernandes Medeiros, Markus Lindström, Naïm Qachri et Alessia Violin.

Édition 2010–2011 (« Big Bang » : passage de C++ à Python 2). Stéphane Fernandes Medeiros, Naïm Qachri et Jérôme Vervier. Relecture par Markus Lindström et Alessia Violin. Remerciements également à Hadrien Mélot (UMons) pour nous avoir laissé nous inspirer des exercices de son cours pour réaliser cette édition.

Les exercices du chapitre « Instructions de base » introduit dans l'édition 2011–2012 ont été inspirés par le syllabus du cours de Programmation (anciennement « Algorithmique et Programmation ») des éditions précédentes, orientées vers le langage C++, auquel ont contribué Nadjet Benseba, Emmanuel Dall'Olio, Martin De Wulf, Gilles Geeraerts, Joël Goossens, Christophe Macq, Olivier Markowitch, Thierry Massart et Patrick Meyer. Cet ouvrage a également servi de base principale pour les chapitres « Complexité » et « Logique et invariants ».

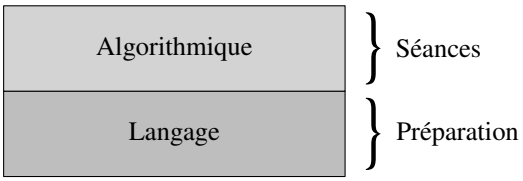
Informations importantes

Les séances d'exercices de programmation servent plusieurs objectifs pédagogiques :
– Maîtriser les bases du langage de programmation Python dans des mises en situation ;
– Développer l'esprit logique et la capacité à réaliser des algorithmes ;
– Mettre en pratique quelques bases d'informatique théorique (complexité, logique et invariants) ;
– Inciter l'étudiant à travailler régulièrement.

Chaque chapitre du présent syllabus est séparé en deux parties :

Préparation. Chaque chapitre exige l'étude préalable de *prérequis* (typiquement, de la matière vue au cours, ou éventuellement vue en secondaire) ainsi que la résolution éventuelle *d'exercices préparatoires*. Ceux-ci forment une base pour la réalisation des exercices en séance. L'objectif de la préparation est de faire acquérir à l'étudiant les éléments nécessaires de *syntaxe* et de *sémantique* du langage de programmation utilisé.

Séance(s). Les exercices en séance sont encadrés par des assistants. Ces derniers constateront si l'étudiant a correctement préparé chaque séance et s'ils peuvent dès lors proposer des problèmes *d'algorithmique* (c'est-à-dire, étant donné les outils fournis par le langage de programmation et leur mise en œuvre, comment peut-on résoudre efficacement un problème donné ?).



Les motivations sous-jacentes à cette découpe sont multiples :
– L'expérience nous a montré que si les étudiants ne maîtrisent pas la syntaxe et la sémantique du langage utilisé, ils ne sont d'une part pas en mesure d'écrire des algorithmes dans ce langage, mais d'autre part ne sont pas non plus armés pour comprendre les explications fournies par les enseignants vu l'absence de vocabulaire commun.
– Les rappels théoriques autrefois donnés en début de TP ne semblaient pas servir d'objectif concret : les étudiants ayant déjà compris la matière perdaient leur temps, tandis que les autres avaient souvent trop de lacunes que pour pouvoir utiliser efficacement l'information fournie. La préparation de la matière avant chaque séance d'exercices devrait pallier ce problème, tout en permettant aux assistants de réduire le temps passé pour des rappels théoriques et d'ainsi pouvoir se concentrer sur la résolution d'exercices et l'acquisition de compétences en algorithmique.

– La préparation individuelle des chapitres confronte l'étudiant à l'interpréteur Python et lui permet de tester son comportement réel, condition *sine qua non* pour la réalisation d'exercices et la réussite du cours.

Les séances sont construites autant que possible pour que les exercices aient une difficulté croissante. Les exercices dénotés par une astérisque * sont réputés de niveau examen, tandis que ceux dénotés par deux astérisques ** sont de difficulté supérieure à ce qui serait demandé à un examen. Ces derniers exercices ont pour but de permettre aux étudiants le souhaitant de se dépasser.

Règles du jeu

L'étudiant est tenu de préparer chaque chapitre de ce syllabus de manière individuelle en s'aidant du cours théorique. Les exercices préparatoires forment un prérequis nécessaire à la compréhension et à la participation aux séances d'exercices. Si l'étudiant rencontre des difficultés qu'il estime insurmontables dans le cadre des exercices préparatoires, il est invité à contacter l'assistant de son groupe par courrier électronique au moins 24 heures avant la séance (compter un jour ouvrable).
Il est demandé à l'étudiant de respecter le travail de ses condisciples aux séances.
Les corrections d'exercices ne seront pas systématiquement fournies, cela afin d'éviter que l'étudiant vienne aux séances uniquement pour recopier les solutions. L'étudiant aura à sa disposition une plateforme interactive d'apprentissage, nommée *UpyLaB*, lui permettant de tester pour chaque exercice fourni son propre programme et d'avoir une indication sur la validité de ce dernier. La plateforme est accessible à l'adresse suivante :

`http://upylab.ulb.ac.be/`

Coordonnées des assistants

L'étudiant peut toujours contacter l'assistant de son groupe en dehors des séances d'exercices en fixant un rendez-vous préalable par courriel. Les coordonnées des assistants sont données par chacun d'eux lors de la première séance et sont également disponibles sur l'Université Virtuelle :

`http://uv.ulb.ac.be/`

Des versions électroniques du présent syllabus d'exercices ainsi que des supports de cours et d'anciens examens sont disponibles à cette même adresse.

Guidances

Des guidances sont à disposition de l'étudiant à partir du mois d'octobre. Toutes les informations relatives à ce service sont disponibles à l'adresse suivante :

`http://www.ulb.ac.be/di/guidances/`

Table des matières

1 Séance introductive **1**

2 Instructions de base **4**

3 Fonctions **10**

4 TurtleWorld **15**

5 Listes **18**

6 Matrices **21**

7 Chaînes de caractères **25**

8 Récursivité **30**

9 Python Imaging Library **32**

10 Dictionnaires **35**

11 Complexité **40**

12 Logique et invariants **45**

13 Tris **51**

14 Fichiers et exceptions **53**

15 Introduction aux classes **56**

16 Base de données **59**

A Aide-mémoire Python **63**

Chapitre 1

Séance introductive

Exercices préparatoires

Matière à réviser :
– les variables ;
– les fonctions `print()` et `type()` ;
– les opérateurs +, −, ⋆, /, //, ⋆⋆, =, +=, −=, ⋆=, /=, //= ;
– les commandes en terminal `ls` et `cd`.
Lisez le *Mode d'emploi introductif pour les salles du NO4 et NO3* partie 1 « Introduction » et la partie 2 « La console Linux » (seulement 2.1, commandes `ls` et `cd`).

Exercices en séance

Ex. 1.1. Allez dans le menu Applications et lancez l'application *Terminal* située dans l'onglet Accessoires. Lancez *Python 3* en tapant la commande `python3` dans le Terminal.

Ex. 1.2. Entrez les expressions suivantes dans l'interpréteur et regardez le résultat :

1. `5`
2. `5 + 1`
3. `x = 5`
4. `x + 1`
5. `x = x + 1`
6. `x`

Ex. 1.3. Utilisez l'interpréteur pour afficher les textes suivants (rappel : `print()`) :

1. `Hello World`
2. `Aujourd'hui`
3. `C'est "Dommage !"`
4. `Hum \0/`

Allez dans le menu Applications et lancez l'application *gedit* (Éditeur de texte) située dans l'onglet Accessoires (alternativement, tapez « `gedit&` » dans le terminal). Créez un fichier avec les commandes que vous avez tapées pour les exercices précédents dans le terminal. Lancez encore un terminal et exécutez le programme que vous venez d'écrire a l'aide de la commande

`python3 nom_de_fichier.py` (positionnez-vous d'abord dans le bon dossier à l'aide des commandes `cd` et `ls`).
Exemple d'un tel fichier :

```
# Mon fichier Python
commande1
commande2
...
commandeN
```

Ex. 1.4. Évaluez (à la main) les expressions suivantes et essayez de deviner le type du résultat. Utilisez ensuite l'interpréteur Python pour vérifier vos réponses (rappel : `type()`). Résolvez les exercices avec divisions en utilisant l'opérateur `/`, puis l'opérateur `//`.

1. 14 - 14
2. 1 + 6.9
3. 1.0 + 2.0
4. $\frac{18}{7+1}$
5. $\frac{(3+2)*2.5}{4*2}$
6. 0 * 0.0
7. $4^{0.5}$
8. $\frac{5}{8}$
9. $-\frac{1}{2}$
10. $3^{-\frac{1}{2}}$

Ex. 1.5. Certaines des lignes de code suivantes contiennent des erreurs. Il peut s'agir d'erreurs syntaxiques ou sémantiques et certaines lignes génèrent des exceptions. Indiquez pour chacune d'entre elles le type d'erreur (s'il y en a) ou le résultat et expliquez brièvement. Vérifiez ensuite à l'aide de l'interpréteur. Le résultat désiré par le programmeur est indiqué en gras.

1. `print (''Bonjour')`
   `Bonjour`
2. `'bla' * 3.0`
   `'blablabla'`
3. `((1 + 4) / (6 * 2)`
   `0.4166666666666667`
4. `int(''14'')`
   `14`
5. `int('3+4')`
   `7`
6. `'3 * 3' * 3 ** 2`
   `81`
7. `3 + 2 / 0 + 2`
   `2`
8. `(print 'Il y a 31 jours en janvier')`
   `Il y a 31 jours en janvier`

9. `print ("Chuck Norris was born in " + 1940)`
   `Chuck Norris was born in 1940`

Ex. 1.6. Résolvez les problèmes suivants en écrivant des petits programmes dans des fichiers séparés. Créez d'abord toutes les variables nécessaires, tapez ensuite la formule en une seul ligne et affichez le résultat :

1. Le volume d'une sphère de rayon r est donné par $\frac{4}{3}\pi r^3$. Quel est le volume d'une sphère de rayon 5 ? Et de rayon 8 ?
2. Le prix affiché d'un livre est de 24.95 €, mais vous bénéficiez d'une réduction de 40 %. Par ailleurs, les frais d'envoi sont de 3 €. Quel est le prix total pour 60 livres ? Quel est le prix total de 50 livres si les frais d'envoi sont de 5 € et que vous bénéficiez d'une réduction de 43 % ?
3. Si vous parcourez 10 kilomètres en 43 minutes et 30 secondes, quelle est votre vitesse moyenne en miles par heure ? Quelle est votre vitesse moyenne en miles par heure si vous parcourez 10 kilomètres en 45 minutes ? Pour rappel : 1,61 km = 1 mile, 1 heure = 60 minutes et 1 minute = 60 secondes.
4. L'édition complète de la série « *Les comptes de Chuck Norris* » est composée de 3486 volumes et se trouve dans ses armoires numérotées dans l'ordre. Si chaque armoire peut contenir au plus 89 livres, dans laquelle se trouve le volume numéro 1024 ? Dans quelle armoire se trouverait le volume 404 si chacune pouvait contenir 91 livres ?

Ex. 1.7. Supposez que vous ayez quatre variables : a, b, c et d. Comment pourriez-vous vous y prendre pour inverser l'ordre des valeurs qu'elles réfèrent sans utiliser l'assignation multiple comme `a,b,c,d = d,c,b,a` ? Par exemple, si à l'initialisation, a = 1, b = "2", c = 3.0 et d = True, comment obtenir `"True 3.0 2 1"` à l'écran en entrant `print(a,b,c,d)` ?

Ex. 1.8. ** Même question que dans l'exercice 1.7 en supposant que les 4 variables sont de type `int`. Vous ne pouvez pas utiliser d'autres variables que `a`, `b`, `c` et `d`.

Ex. 1.9. Allez sur le site Python.org et cherchez dans la documentation du langage (« language reference ») de l'aide à propos de `print`. Comparez avec le résultat obtenu dans l'interpréteur lorsque vous entrez `help('print')`. Cherchez à vous renseigner sur d'autres aspects de Python vus au cours (`int`, `str`, les opérateurs, ...).

Chapitre 2

Instructions de base

Exercices préparatoires

Matière à réviser :
– les variables ;
– l'indentation ;
– les fonctions prédéfinies `input()`, `print()`, `range()` ;
– les types `int`, `float`, `bool` ;
– les opérateurs de comparaison et connecteurs logiques (>, <, ==, `and`, `or`, etc.) ;
– les instructions de branchement conditionnel `if-else`, `if-elif-else` ;
– les boucles `for` et `while`.
Faites tous les exercices préparatoires sur papier et ensuite vérifiez vos réponses à l'aide de l'interpréteur `python3`.

Prép. 2.1. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* les valeurs 2 et 6 ?

```
a = int(input())
a *= 2
b = int(input())
b += a
print(a)
print(b)
```

Prép. 2.2. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* les valeurs 2, 6 et 4 ?

```
a = int(input())
b = int(input())
a = b
b = int(input())
b += a
print(a,b)
```

Prép. 2.3. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* les valeurs 2 et 6 ?

```
b = int(input())
a = int(input())
a = b+1
```

```
print(a)
a = b+1
print(a)
a += 1
print(a,a+1)
```

Prép. 2.4. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* :

1. les valeurs 2 et 6 ?
2. les valeurs 8 et 3 ?
3. les valeurs 3 et 3 ?

```
a = int(input())
b = int(input())
if a > b:
  print(a)
  a = b
print(a)
```

Prép. 2.5. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* :

1. les valeurs 2 et 6 ?
2. les valeurs 8 et 3 ?
3. les valeurs 3 et 3 ?

```
a = int(input())
b = int(input())
if a > b:
  print(a)
  a = b
  print(a)
```

Prép. 2.6. Donnez la valeur des variables `a`, `b`, `c`, `arret`, `test1`, `test2` et `test3` après chacune des instructions ci-dessous :

```
a = 2
b = 3
c = 4
test1 = True
test2 = (b>=a) and (c>=b)
test3 = test1 or test2
arret = test3 and (not test2)
a += 1
b -= 1
c -= 2
test1 = True
test2 = (b>=a) and (c>=b)
test3 = test1 or test2
arret = arret or test2
```

Prép. 2.7. Qu'écrit sur *output* le programme suivant quand on lui fournit en *input* :

1. les valeurs 1 et 6 ?

2. les valeurs -8 et 2.5 ?

```
a = float(input())
b = float(input())
test = (a >= b)
if test or a < 0:
  print("Oui!")
else:
  print("Non!")
```

Prép. 2.8. Que fait cette suite d'instructions ?

```
a = 1
while a <= 5:
  print(a, end=' ')
  a += 1
print(a)
```

Prép. 2.9. Que fait cette suite d'instructions ?

```
a = 1
while a < 5:
  a += 1
  print(a, end=' ')
print(a)
```

Prép. 2.10. Que fait cette suite d'instructions ?

```
i = 0
for j in range(10):
  i += j
print(i)
```

Prép. 2.11. Quelle est la différence entre les quatre instructions suivantes ? Expliquez ce qu'elles font et essayez de les utiliser avec une boucle `for`.

```
range(42)
range(42,69)
range(42,69,3)
range(42,-69,-1)
```

Exercices en séance

Ex. 2.1. Écrire un programme qui affiche a^{17} (a lu sur *input*) en employant le moins de multiplications possibles, sans utiliser l'opérateur $\star\star$ ni de boucle.

Ex. 2.2. Écrire un programme qui affiche $18 \times a$ (a lu sur *input*) en employant le moins d'instructions possibles, sans utiliser l'opérateur $\star$ ni de boucle.

Branchements conditionnels

Ex. 2.3. Écrire un programme qui lit 3 nombres, et qui, si au moins deux d'entre eux ont la même valeur, imprime cette valeur (le programme n'imprime rien dans le cas contraire).

Ex. 2.4. Pour chacune des 4 instructions d'affichage (`print`), donner l'ensemble des valeurs de a pour lesquelles celles-ci seront exécutées.

```
a = int(input())
if a > 0:
  if a > 1:
    if a > 2:
      print(a-2)
    else:
      print(a-1)
  else:
    print(a)
else:
  print("Erreur")
```

Ex. 2.5. Écrire le morceau de code qui si, a est supérieur à 0, teste si a vaut 1, auquel cas il imprime « a vaut 1 » et qui, si a n'est pas supérieur à 0, imprime « a est inférieur ou égal à 0 ».

Ex. 2.6. Écrire le programme qui lit en input trois entiers a, b et c. Si l'entier c est égal à 1, alors le programme affiche sur output la valeur de $a + b$; si c vaut 2 alors le programme affiche la valeur de $a - b$; si c est égal à 3 alors l'output sera la valeur de $a \times b$. Enfin, si la valeur 4 est assignée à c, alors le programme affiche la valeur de $a^2 + b \times a$. Si c contient une autre valeur, le programme affiche un message d'erreur.

Ex. 2.7. Écrire un programme qui imprime la moyenne arithmétique ($\frac{a+b}{2}$) de deux nombres lus sur input.

Ex. 2.8. Écrire un programme qui imprime la moyenne géométrique ($\sqrt{ab}$) de deux nombres lus sur input.

Ex. 2.9. Écrire un programme qui lit deux nombres a et b sur input et qui calcule et affiche le nombre c tel que b soit la moyenne arithmétique de a et c.

Boucles simples

Ex. 2.10. Écrire l'équivalent des instructions suivantes avec une boucle `while` :

```
for i in range(a,b+1):
  print(i)
```

Ex. 2.11. Remplacez la boucle `while` une boucle `for` dans le code ci-dessous :

```
i = 1
while i < 42:
  print(i)
  i += 1
```

Ex. 2.12. Pour cette exercice vous ne pouvez pas utiliser d'instruction `if`. Écrire un programme qui lit un nombre n sur input et qui affiche tous les nombres naturels :

1. entre 0 et n compris, de manière croissante (utilisez la boucle `for`) ;
2. entre 0 et n compris, de manière décroissante (utilisez la boucle `while`) ;
3. pairs entre 0 et n compris, de manière croissante (utilisez la boucle `while`) ;

4. entre 0 et n, bornes non comprises, de manière croissante (utilisez la boucle `for`) ;
5. multiples de 7 entre 0 et n bornes non compris, de manière croissante (utilisez la boucle `for`) ;
6. multiples de 5 entre 0 et n compris, de manière décroissante (utilisez la boucle `for`) ;

Ex. 2.13. Modifiez le code ci-dessous en utilisant une boucle `for` :

```
i = 1
while i < 42:
  print(i)
  i = 2*i
```

Ex. 2.14. Écrire un programme qui calcule la moyenne des résultats d'une interrogation, ces notes étant données en input, la fin des données étant signalée par la valeur sentinelle -1 (on suppose aussi que la suite des notes contient toujours au moins un élément).
Exemple d'exécution :

```
Entrez les notes.
Note 1 : 2
Note 2 : 8
Note 3 : 14
Note 4 : -1
La moyenne est de 8
```

Ex. 2.15. Écrivez un mini jeu : le programme choisit un nombre naturel (nombre secret) entre 0 et 100 de manière aléatoire, le jouer doit deviner ce nombre en utilisant le moins d'essais possibles. A chaque tour le joueur peut faire un essais et le programme doit répondre si le nombre secret est plus grand ou plus petit. Si le joueur trouve le nombre secret, alors le programme doit afficher `"Bravo !"` ainsi que le nombre d'essais utilisés par le joueur.
Le joueur a maximum 7 essais ; s'il ne devine pas le secret après 7 essais, le programme s'arrête en écrivant `"Vous avez perdu !"`.
Pour générer des nombres aléatoires importez le module `random`. Pour générer un entier aléatoire dans intervalle $[0; n]$ utilisez la fonction `randint(0,n)`.
Exemple d'exécution :

```
J'ai choisi un nombre secret !
Quel est le secret ? 50
Non, le secret est plus petit !
Quel est le secret ? 25
Non, le secret est plus grand !
Quel est le secret ? 37
Bravo !
Le secret (37) est trouvé en 3 essais.
```

Ex. 2.16.* On peut calculer approximativement le sinus de x en effectuant la sommation des n premiers termes de la série (c'est-à-dire la somme infinie)

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

où x est exprimé en radians.
Réécrivez cette somme sous la forme

$$\sin(x) = \sum_{i=0}^{\infty} f(i, x)$$

On vous demande donc de trouver $f(i, x)$. Écrivez ensuite le code calculant de cette manière la valeur de $\sin(x)$ où x est lu sur input. Continuez l'addition des termes successifs dans la série jusqu'à ce que la valeur d'un terme devienne inférieur (en valeur absolue) à une constante ε (par exemple, $\varepsilon = 10^{-5}$, ce qu'on noter `1e-5` en Python).

Boucles imbriquées

Ex. 2.17. Écrire un programme qui lit sur input une valeur naturelle n et qui affiche à l'écran un carré de n caractères `X` de côté, comme suit (pour $n = 6$) :

```
X X X X X X
X X X X X X
X X X X X X
X X X X X X
X X X X X X
X X X X X X
```

Ex. 2.18. Variante de l'exercice 2.17, afficher le triangle supérieur droit, comme suit (pour $n = 6$) :

```
X X X X X X
  X X X X X
    X X X X
      X X X
        X X
          X
```

D'autres variantes :
– afficher uniquement le bord du carré ;
– afficher le triangle inférieur gauche (supérieur gauche, inférieur droit) ;
– afficher un rectangle.

Ex. 2.19. Refaire l'exercice 2.17 en supposant que n est impair et en dessinant des `O` sur les deux diagonales principales à la place des `X`. Par exemple, pour $n = 7$:

```
O X X X X X O
X O X X X O X
X X O X O X X
X X X O X X X
X X O X O X X
X O X X X O X
O X X X X X O
```

Ex. 2.20. Refaire l'exercice 2.19 sans utiliser de branchement conditionnel (pas de `if-else`).